DATA BASE MANAGEMENT SYSTEM FOR ACCOUNTING

5

12127CH05

Learning Objectives

After studying this unit you will be able to:

- *Understand how to Structure database as per requirement. Design and create database tables.*
- *Make use of Microsoft Access for simple database applications involving creation of back-end database and the front-end forms for capturing, processing and retrieval of data.*

Introduction

One of the major factors for realising the need of Computerised Accounting System is the overwhelming quantity of data in our organisations. The conventionally used paper filing system, text documents, and even spread-sheets may not suffice for the growing needs of tracking this voluminous and critical information. A simple solution to this situation is available in the form of a *Database Management System* (DBMS) (e.g. 'Access', 'Oracle', 'SQL Server', etc.) that provides a variety of software tools for organising, processing and querying data in a flexible manner.

As we now proceed to look into practical applications of computers in generating, storing, processing, and retrieving of accounting information, we will make an assumption that you are fairly conversant with the *accounting framework* and *operating procedure* – i.e. you have the required domain knowledge. We also assume that you have adequate exposure to handling of computers and the concepts of database. However, we do not wish to make knowledge of the *database management* a pre-requisite for understanding of this chapter, and hence, we will restrict ourselves to the simpler and easy to comprehend 'MS Access' program for developing some practical accounting applications. In doing so, we will focus on the three major components of Access, namely, '**tables**', '**queries**' and '**forms**'. Having done that, we will examine the methods of generating **reports**. This basically builds on the concepts of **DBMS** already learnt by you in class XI.

5.1 UNDERSTANDING AND DEFINING THE DATABASE REQUIREMENT

With the continuous improvements in computer's processing speed, storage capacity, networking techniques, operating systems, etc, the capabilities of computer applications have also gone up many folds. Various computer applications that are commercially available today not only provide fairly comprehensive tools for all conceivable needs but also have become extensively user friendly. So, when we look forward to putting into use database applications such as 'Access', we really do not need much of programming skills. Nevertheless, any programming knowledge may improve our efficiency and effectiveness in handling such applications. On the other hand, before we develop any database application, we ought to have a complete understanding of our requirements expected from the application. This is one area where application itself may not extend much help. Further, the correct understanding of our requirement also has a bearing on the choice of *Database Management Systems* (DBMS) - i.e. whether to go for a 'desktop database' or to choose the 'server database'.

> ***Box - 5.1***
>
> *In most of the cases, database is not directly accessible to users. Any addition, modification or retrieval of information from database is done by the user-friendly programs. Database is thus rightly referred to as '**back-end**' while the interactive program is termed as '**front-end**' of a database application.*

While the *desktop databases* – residing on standard personal computers – are oriented toward single-user applications, 'server databases' are geared toward multi-user applications. Understandably the 'server database', containing additional provisions for ensuring reliability and consistency of data in a multi-user environment, are substantially costlier than the 'desktop database'. Hence, it is imperative to do a careful analysis of our requirements before investing in a database solution. Some of the questions that need to be answered in this regard are:

- What all data is required to be stored in the database?
- Who will capture or modify the data, and how frequently the data will be modified?
- Who all will be using the database, and what all tasks will they perform?
- Will the database (backend) be used by any other front-end application?
- Will access to database be given over LAN (*Local Area Network*)/ Internet, and for what purposes?
- What level of hardware and operating system is available?

If our requirement entails approach to database by many users, or if it involves simultaneous updating of data by different users, then perhaps *server databases* such as 'Microsoft SQL Server', 'Oracle' or **'IBM DB2'** will provide desired solutions.

Although carrying high price tags, a server-based database can give you several advantages in terms of flexibility for the choice of front-end applications, powerful performance independent of platform, and scalability to handle rapidly expanding number of users as well as amount of data.

Box - 5.2

*Quite often, same database is required to be used by different applications. For example, an organisation may have automated attendance recording system which will record the arrival and departure time of employees. A variety of such systems (e.g. Card-swipe, Biometric, etc.) are available in the market, and all of them come with their own (back-end) database and requisite (front-end) programs. This attendance information may be required for generating salary bills of the employees. The payroll application of the organisation, with its own **(back-end)** database and **(front-end)** program, will also access the database of attendance recording application.*

Comparatively, the *desktop databases* offer an inexpensive and simple solution to many of our business data storage and processing requirements. The 'Microsoft Access' is one such database - which comes with the licensed copy of Microsoft Office Professional – providing one of the simplest and most flexible DBMS solutions today, besides giving the advantages of familiar 'Windows' look and integration with other Microsoft Office products such as 'Excel'.

Before opting for any of the *database management system*, it will be necessary to ensure that your hardware as well as operating system meets the minimum system requirements of the desired DBMS**.** As stated earlier, we will be enumerating examples of 'Access' applications, and hence it is imperative that your computer lab is equipped with the licensed copy of 'MS Access' with compatible configuration of the hardware. The illustrations contained in this book are based on Access-2007, which may feel a little different from what you are working with in case your application is of a different version.

Box - 5.3

Access 2007 usually comes as part of Microsoft Office 2007. However, it can also be purchased separately as a standalone program. In order to run this program, your computer should have a minimum configuration of 500 MHz Processor, 256 megabyte RAM, 1.5 gigabyte Hard disk, and Windows XP Operating system.

Even on same version of the 'Access', marginal differences may appear in the illustrated diagrams and the views on your computer and this may happen on account of variations in the customised settings of the application. However, these variations are not expected to hamper your understanding of the contents of this chapter.

5.2 IDENTIFICATION OF DATA TO BE STORED IN TABLES

By now we understand that a database is simply an organised collection of data with 'tables' as its fundamental building blocks. Tables allow us to create the framework for storing information in the database. Each column (also called '**field**') of the table corresponds to a specific characteristic (or '**attribute**' in database terms) of the stored information. Each row (also called '**record**') corresponds to a particular instance of the information.

Box - 5.4

In Access 2007, the entire database is encompassed in a file (with extension .accdb) which can be stored in your hard-drive or CD. There can be multiple tables (each storing a specific set of data) with in this file. Then there are multiple fields in each table according to different categories (types) of data within the table. And at every instance (occurrence) of a collection of data covering all fields, a row/record gets created.

A set of tables often with well established relationships between them constitutes the database covering total spectrum of stored information. The term '**database design**' can be used to describe the structure of different parts of the overall database.

Box - 5.5

*Identification of various attributes of a database is generally considered as part of **requirement analysis.** Such a process may entail database designer to elicit needed information from those with the domain knowledge.*

We may get a better understanding of database design by referring to practical example of an accounting problem. We may take up the case of *payroll accounting*, concepts of which have already been examined while studying spreadsheet applications in the earlier chapter(refer chapter 3). You have already seen how different employees draw different amounts of pay based on the nature and levels of their employment, even while subjected to a definite pattern of 'pay rules' prevalent in the organisation. Since pay rules vary widely from organisations to organisations, the database design for them will also

differ considerably. For our illustration in this chapter, we will consider the simplified pay pattern for an assumed organisation XYZ Pvt. Ltd.

Let us begin by identifying various attributes of information that are required to be stored in our Payroll database. We may have a set of attributes pertaining to employee's personal details such as: 'Employee ID', 'Name', 'Designation', and 'Location'. On the other hand we may also deal with such attributes pertaining to employee's pay as: 'Basic Pay', 'Dearness Allowance (DA)', 'House Rent Allowance (HRA)', 'Transport Allowance (TA)', 'Provident Fund (PF) Deduction', etc. We may also want to know the attributes of 'Gross Salary' and 'Net Salary' which is obtained by subtracting 'PF Deductions' from 'Gross Salary'. However, 'Gross Salary' and 'Net Salary' attributes may not require being stored in the database as they are merely computational outcomes from other attributes. We may also require some attributes concerning pay formulations such as '% Rate of DA' which may fluctuate month to month, '% Rate of HRA' varying with the location of employee, and 'TA Slabs' varying with the designation of employee.

Box - 5.6

At this stage you may like to know the limitations of 'Excel' in payroll accounting. If you look back at payroll example in Excel spreadsheet, you will notice that you have created pay information of a set of employees for a particular month. For generating information concerning next month, you will have to change employee specific data such as 'Actual Basic Pay', 'days of attendance', etc in respective cells. However, the moment you do this, the information of previous month will be lost. Understandably, 'Excel' is merely providing a template for certain calculations concerning salaries, and storing single (monthly) instance of the derived information. Excel eliminates redundant data and hence multiple instances of information cannot be stored in a single Excel sheet. You may avoid this situation by effecting changes in a different copy of the sheet. In this manner you are creating different spread sheet for different months. This is a cumbersome task, and furthermore the information in such a case will not be residing in a single table, or a set of related tables, from where it may be retrieved for generation of 'annual income' or 'income tax' information.

Now the main question is how to store these attributes in our database tables. Shall we make all of above attributes as part of one table, or shall we opt for multiple tables. The 'Access' as such will not put any constraint on the number of tables we opt for, or the type of data we chose to put in any table. It will have to be entirely our decision, based on logical structuring of data that we seek to apply. You may also apreciate the fact that the purpose of a database is not so much for the storage of information, as for its quick retrieval. Hence, you ought to structure your database in such a manner that it can be queried quickly and efficiently.

5.3 LOGICAL STRUCTURING OF DATA IN TABLES

Suppose, we seek the simplest structure of a single table containing all attributes listed above. The rows of this table (see Figure-5.1) will contain both employee's personal details as well as employee's pay details for every instance of the salary drawn by him or her. Since, the information will be generated on monthly basis, multiple records will be created for each employee in the table. Understandably for any particular employee, we may expect variations in the pay attributes recorded under different rows - i.e. different pay components and the resulting gross/net salary may vary from month to month. However, we do not expect monthly variations in the personal attributes of the same person under different rows. Evidently thus, the structure of single table may not be an efficient one as it will entail unwarranted recording of the same personal data multiple times.

Month & Year	Emp Name	Desig.	Locat.	%DA Rate	%HRA Rate	Basic	DA	HRA	TA	Gross Salary	PF Ded.	Net Salary
Nov. 2007	Ram Kishore	Chief Manager	Delhi	26	30	25000	6500	7500	7000	46000	5000	41000
Nov. 2007	Kishan Sharma	Manager	Faridabad	26	20	22000	5720	4400	5000	37120	3000	34120
Nov. 2007	Rupali Varma	Senior Engineer	Meerut	26	15	20000	5200	3000	3500	31700	2000	29700
Nov. 2007	Surjeet Singh	Engineer	Meerut	26	15	16000	4160	2400	3500	26060	2000	24060
Dec. 2007	Ram Kishore	Chief Manager	Delhi	26	30	25000	6500	7500	7000	46000	7000	39000
Dec. 2007	Kishan Sharma	Manager	Faridabad	26	20	22000	5720	4400	5000	37120	3000	34120
Dec. 2007	Rupali Varma	Senior Engineer	Meerut	26	15	20000	5200	3000	3500	31700	3000	28700
Dec. 2007	Surjeet Singh	Engineer	Meerut	26	15	16000	4160	2400	3500	26060	3000	23060
Jan. 2008	Ram Kishore	Chief Manager	Delhi	30	30	25000	7500	7500	7000	47000	6000	41000
Jan. 2008	Kishan Sharma	Manager	Faridabad	30	20	22000	6600	4400	5000	38000	2000	36000
Repeating personal data				Pay formulation parameters								Pay data

Figure 5.1: Single table for all attributes

With the understanding of the limitations of a single table, let us now examine a better approach involving creation of multiple tables. For example, we may have separate tables for personal attributes (i.e. name, designation, location etc.), pay formulation parameters (i.e. % DA rate, % HRA rate, etc.) and the pay attributes (i.e. basic pay, DA,

HRA, TA, etc.). Personal details table may be used for recording personal attributes of every employee, consuming only one row of data (i.e. record) for each employee. The pay details table may contain records of pay attributes for all employees, with a new row of data created for an employee for every instance of the monthly salary drawn by him or her. Other tables may also be created for recording different parameters used in pay formulations of employees. Thus, we may have separate tables for storing the '% DA rate' for different months, '% HRA rate' for different locations, and the fixed quantum of 'Transport Allowances' for different designations of employees.

In order to establish relationship between any two tables, we will have to insert columns with matching values in the two related tables. Thus, we may have 'employee name' appearing in both 'personal table' as well as in 'pay table'. After establishing a relationship between these two tables, rows with a common value in 'employee name' field of both the tables get related. The two common fields used in a relationship between tables are called the **key fields**. In our example, any 'employee name' in the personal table appears only once (i.e. unique row); and hence this field of personal table is a **primary key** of the relationship. On the other hand, an 'employee name' in the pay table appears in multiple rows; and hence this field of pay table is a **foreign key** of the relationship. The Figure-5.2 below illustrates the relationship of some such tables, wherein the lines connecting rows of two tables represent relationships.

Emp Name	*Desig.*	*Locat.*
Ram Kishore	Chief Manager	Delhi
Kishan Sharma	Manager	Faridabad
Rupali Varma	Senior Engineer	Meerut
Surjeet Singh	Engineer	Meerut

Locat.	*% HRA Rate*
Delhi	30
Faridabad	20
Meerut	15

Month and Year	*% DA Rate*
Nov. 2007	26
Dec. 2007	26
Jan. 2008	30

Month & Year	*Emp Name*	*Basic*	*DA*	*HRA*	*TA*	*Gross Salary*	*PF Ded.*	*Net Salary*
Nov. 2007	Ram Kishore	25000	6500	7500	7000	46000	5000	41000
Nov. 2007	Kishan Sharma	22000	5720	4400	5000	37120	3000	34120
Nov. 2007	Rupali Varma	20000	5200	3000	3500	31700	2000	29700
Nov. 2007	Surjeet Singh	16000	4160	2400	3500	26060	2000	24060
Dec. 2007	Ram Kishore	25000	6500	7500	7000	46000	7000	39000

Figure 5.2: Multiple tables for different attributes

Box - 5.7

Avoiding duplication of information is key criteria of database design, which is achieved by breaking up of information into separate but related tables; and this process is called ***normalisation****. We will also have to establish links between different tables so as to reconstruct the original information; and these links in database terms are called* ***relationships****. The database created on the basis of such relationships between different data tables is called* ***relational database****.*

Box - 5.8

The process of matching rows in two tables based on their primary and foreign keys is called a ***Join****. Joins along with* ***Structured Query Language*** *(SQL) serve as valuable tools for manipulating tables. But, these topics are beyond the scope of this book, and hence not elaborated further.*

For our practical example, let us take the multiple-tables route. We will have altogether five tables. Our first table, with the name '**TabEmpDetails**', will store employee's personal attributes in the fields of '**EmpName**', '**Designation ID**' and '**Location ID**'. In addition we will also have a field of 'EmpID' for storing unique values of employee number. So this field will be identified as **Primary Key** for the 'TabEmpDetails' table; and thereby database will not permit storing of two similar values in this field. The 'DesignationID' field will merely store the ID of employee's Designation, further details of which will be stored in a separate table. Understandably, this field will have multiple repetitions of some of the values. This field will form one of the **foreign keys** of 'TabEmpDetails' table with the establishment of its relationship with the table containing details of Designations. The 'LocationID' field will get similar treatment with another table containing details of Locations.

You may notice that we have used combinations of shortened words for identification of table name and the names of its different fields. This is done so as not to make headings of the table unnecessarily long, at the same time leaving sufficient scope for easy identification of intended attributes. You may also mark that no blank (gap) is left in between a name. We have also used both upper and lower cases for improving our readability of these names, though Access program as such is case insensitive. Read Box-5.9 to know more about the naming convention.

Our second table will be named '**TabDesignations**' containing fields of '**DesgID**', '**Designation'**, and '**TA'**. Evidently, the field of '**DesgID'** will form the Primary Key for this table. The third table with the name of '**TabLocations'** and containing fields of '**LocationID'**, '**Location'** and

Box - 5.9

For writing a field name you can use up to 64 characters including letters, numbers and blank spaces. However it is advisable to avoid blanks as it entails irritating and unnecessary use of additional identifiers while using names in SQL and programming components (e.g. "Emp ID" will be written as [Emp ID] or Emp_ID, whereas it could have been easily written as EmpID in both database as well as programming). A name cannot start with a blank. You can include punctuation characters except a period (.), an exclamation mark (!), or brackets ([]). You cannot use a field name twice in the same table. It is also advisagle to avoid use of standard words that define Access functions and properties or are part of VB language. Same rules apply for naming objects like Tables, Queries and Forms; however, no two objects of the same type can have same name.

'**RateOfHRA**' will have 'LocationID' as its Primary Key. Our fourth table will be used for storing the percentage rate of DA for different months of salary and it will be named as '**TabDARates**'. It will have fields of '**MonthID**', '**SalMonth**' and '**RateOfDA**' with the field of 'MonthID' forming the Primary Key. Now our last table will be used for storing the salary information that will be generated for different employees in different months. This table, named as '**TabMonthlySalary**' will contain fields of '**SalaryID**', '**MonthID**', '**EmpID**', '**Basic**', and '**DedForPF**' (Deduction for Provident Fund). The field of 'SalaryID' will be used for making each salary record as unique data and hence it will form the Primary Key of the table. The 'MonthID' (forming relationship with 'TabDARates') and 'EmpID' (forming relationship with 'TabEmpDetails') will be the two foreign keys of this table. In this table, the field of 'Basic' will be entered by the Pay-clerk for every instance of salary generation for an employee, depending upon the level of employee and his or her attendance record for the month. The Pay-clerk will also record the amount of PF deduction based on the choice given by individual employee, subject to prevalent rules. You may notice that out five tables, only the 'TabMonthlySal' table will grow every month with the addition of as many records as the number of employees drawing salary for the given month.

By now, you may have noticed that we have not given any fields in any of the table for storing attributes of 'DA', 'HRA', 'TA', 'GrossSalary' and 'NetSalary'. These fields are computed fields; and as such the Pay-clerk is not required to enter any data for these fields. Then why store them unnecessarily. Rather we will generate these fields in a **Query**, which is another Access object designed to extract data form one or more tables. Queries also allow computational facilities and present the outcome in a tabulated manner, as we shall see in a short while.

5.4 CREATING DATABASE TABLES IN MICROSOFT ACCESS

Before we take up the task of database design using Access, we will have to first start up the Microsoft Access Application. For this, click on the ***Start*** button on the Windows Taskbar. Now point to ***All Programs***; then point to ***Microsoft Office***; and then click ***Microsoft Office Access 2007***. (Henceforth in this chapter we will notify such sequential operations in a simplified manner as: ***Start > All Programs > Microsoft Office > Microsoft Access 2007***). You will be taken to the 'Getting Started with Microsoft Access' Screen as shown in Figure-6.3. This screen is divided into three sections. The '*Template Categories*' section on the left is for previewing and downloading standard database templates. In the centre you will see the '***New Blank Database***' section for starting straightaway with a new database. On the right side is the '***Open Recent Database***' section for opening the existing database files.

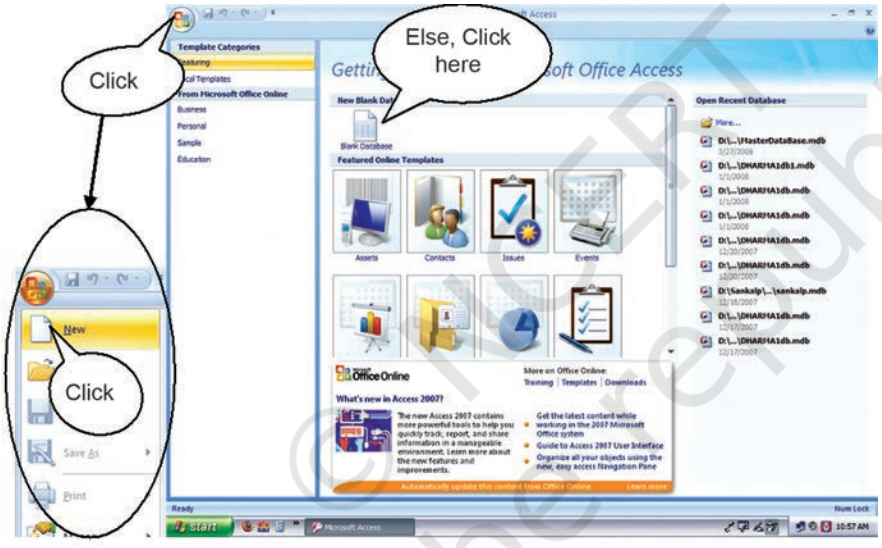


Figure 5.3: Getting started with Microsoft Office Access

Click on ***Blank Database*** under central section. This will open up a dialogue box on the right section asking for database file name. This dialogue box could have also been opened by Clicking on the ***Office button*** located at the upper left corner of the screen and then Clicking ***New*** at the drop down menu (see Figure - 5.3). Enter the file name '**PayRollApplication**' and then click on the ***Create*** button as shown in Figure – 5.4.

Box - 5.10

*If you have so far worked only with Microsoft Office 97-2003 or its earlier versions, then the window style that you may find now may look little unfamiliar. However, this style is now common in all Microsoft Office 2007 applications. Herein Microsoft makes use of a new results-oriented user interface through which commands and features that were often buried in complex menus and toolbars are now easier to find. The user interface now includes task-oriented **tabs** that contain logical groups of commands and features in a standard area called **Ribbon**.*

Instead of the default location for your database file, you can also identify your own location by clicking on the Browse icon given on the right side of file name. In New File dialogue box, you can also accept the suggested name (in default folder My Documents) and later on rename the file and move it to any other folder of your choice. File name can be of any length, and it can include spaces and even most of the punctuation marks. Also note that even if you do not type the default extension "*.accdb*" with the file name, it will be automatically added by the application.

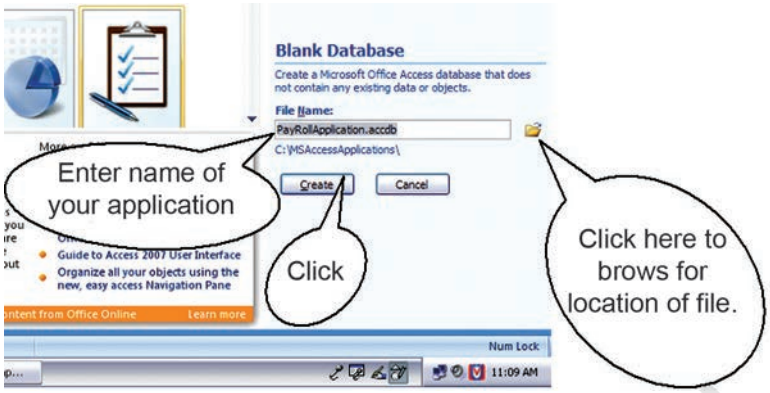


Figure 5.4: Dialogue Box for Creating New Database File

Box - 5.11

'Access' as such is a fully complete application development system that includes Visual Basic (VB) Programming language and several other tools for setting up sophisticated applications. Even if you do not know many of the programming concepts, you can still maximise use of Access with the help of Access ***Wizards*** *that will guide you through several steps for automatic creation of a database of your choice as well as linked queries and forms. For better understanding of underlying concepts, we will mostly use the design tools, and in some cases the route of Wizards as well as.*

Clicking ***Create*** button on the New file dialogue box takes you directly to the new database window with the ***Tab*** for ***DataSheet*** opened up. The Figure 5 illustrates different components of this active database window of Access. In the left side ***Navigation Pane*** you will see that a default table (Table 5.1) has already been created and the same is also opened up in the working area of the window.

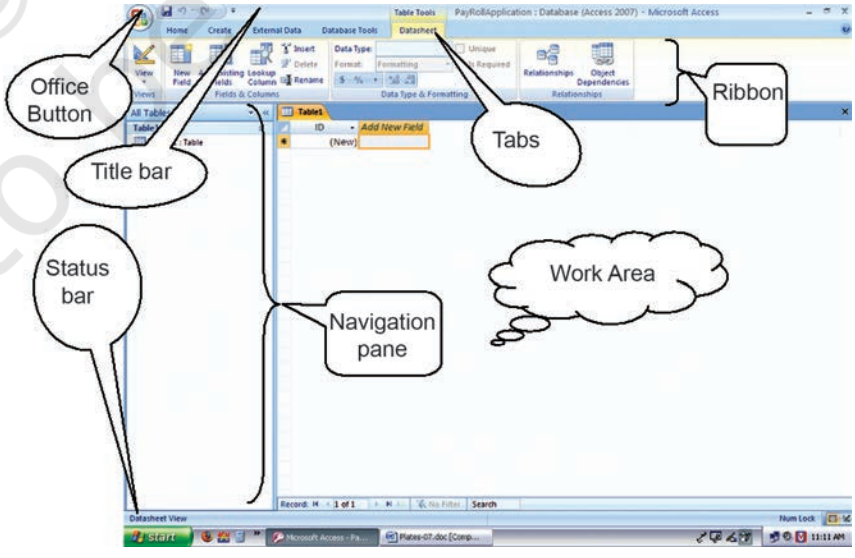


Figure 5.5: Illustration of the Active Database Window

We can straight away start working on this table assuming it to be our first table. All we have to do is to type an employee name (say 'Ram Kishore') in the second column and click ***enter*** (or use ***Tab***) to move to third column; type a designation ID (say '1') followed by ***enter;*** and finally move to fourth column and type a location ID (say '1') - as illustrated in Figure - 6. In this case, Access will automatically set a data type for each field based on the type of data entered into each column. You will see that headings of these colums are named as 'Field1', 'Field2' and 'Field3'. Move the mouse pointer over the column heading 'Field1' and then double-click to select the column heading. Type 'EmpName' and press enter to change the column name from 'Field1' to 'EmpName'. Similary, change the names of columns 'Field2' and 'Field3' to 'DesgID' and 'LocationID'. You might have noticed by now that the first column of the table has the name 'ID' which was created automatically. This is an AutoNumber field in which the field value is assigned automatically by Access as we enter a new record. You may see your first record starting with a number different from the expected '1'. Even this number will be replace by the next number if you delete the record and start afresh. Ignore this limitation as it may not have any impact on our application. Move the mouse pointer over 'ID' heading; double click to select this heading, and change its value to 'EmpID'.

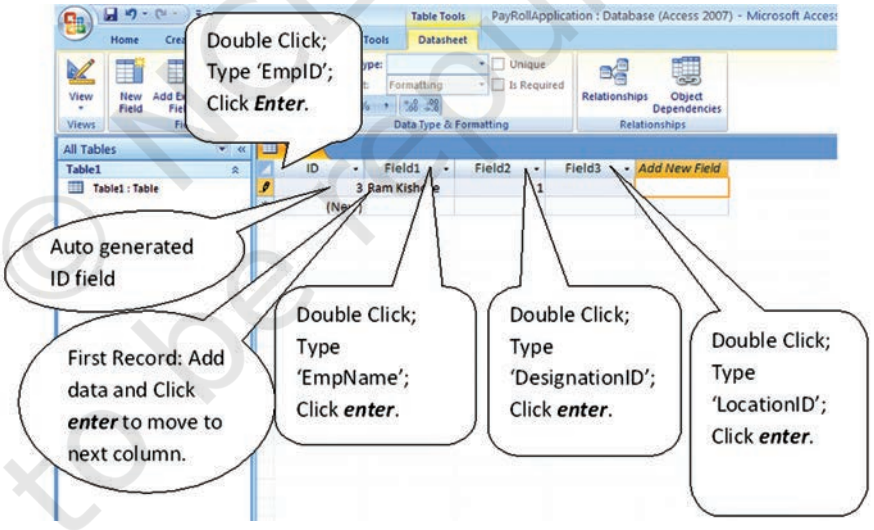


Figure 5.6: Creating a Table by Adding Records

The structure of the 'Table1' that you see presently in the **Datasheet View** is already complete. At this stage, you can chose to enter the values of the other records as well. If you are not contended with the width of any column in your table then with the insertion point positioned in any record of this column click ***Home Tab > More (under Records Group) > Column Width***. In the dialogue box that appears with the present value of column width, you can enter a different value and click Ok; else click BestFit to set the width of column to

accommodate the longest entry (See Figure 5.7). You can also do this operation in a simple manner using mouse while in Datasheet View. Position the pointer on the right column boundary of the selected column in the header row till the pointer changes to a vertical line with a left-right-pointing arrow; then double click the left mouse-button to get the best fit.

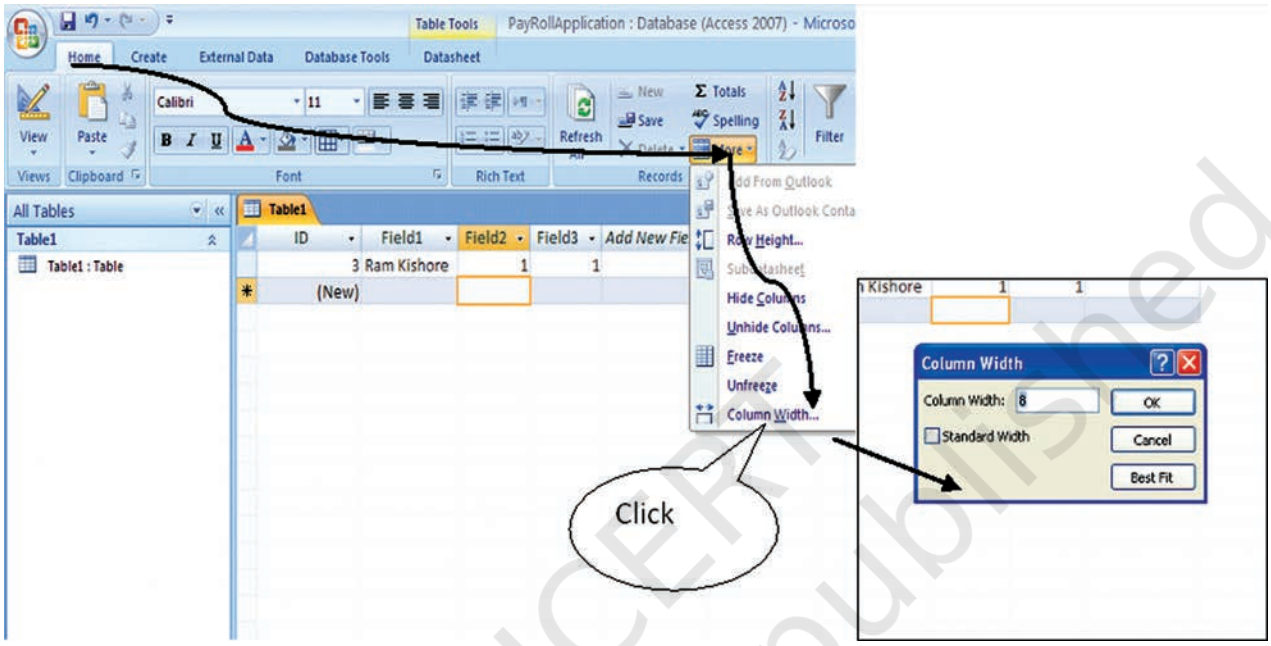


Figure 5.7 : Column width adjustment

Did you notice that our first table is still named 'Table1' and it has not been saved yet. Click on the ***Save*** button on the ***Quick Access*** toolbar on left hand top corner of window. A ***Save As*** dialogue box will

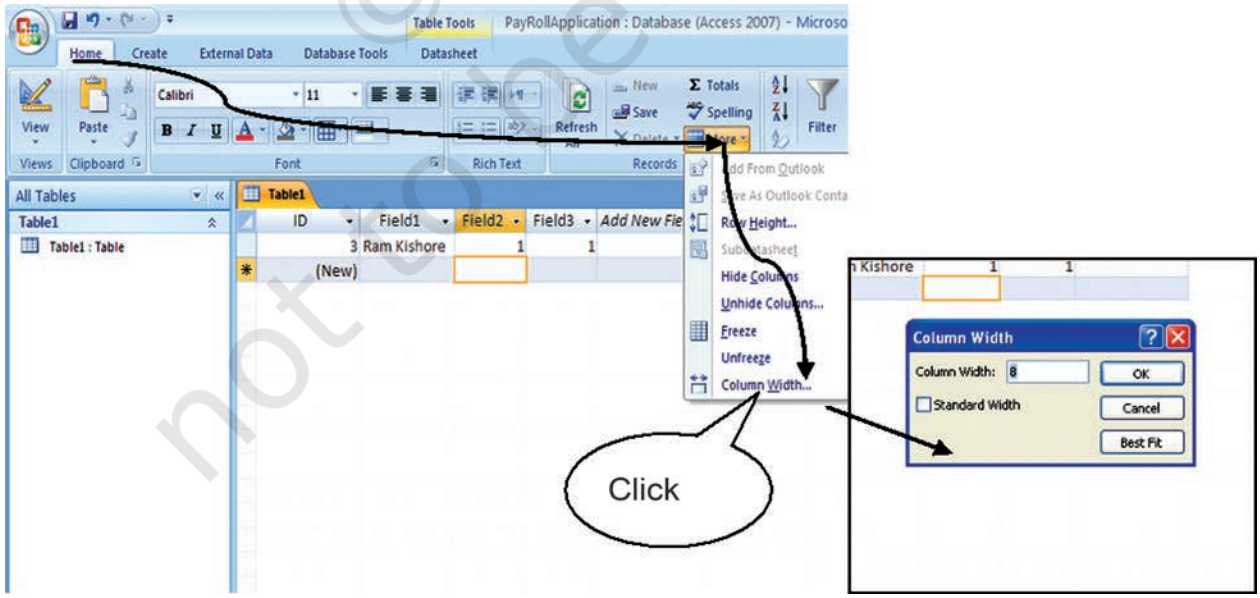


Figure 5.8 : Saving of the Table with intended Name

appear with the default table name of 'Table1' as shown in Figure - 5.8. Replace the name with '**TabEmpDetails**' and press ***enter*** or clik ***OK*** button to save the table with the intended name.

Now let us look at this table in **Design View**. Click on the **View** button placed on the left end of the ribbon (in View group) under the Datasheet Tab as indicated in Figure - 5.8. The design view of the current table will appear in the working area of the window as shown in Figure – 9. In Design View, you can tell Access which all fields will go into the table by entering desired attributes in the column titled 'Field Name'. The type of data corresponding to each attribute can be identified in the column 'Data Type'. Presently in our case for 'TabEmpDetails' table, the field name as well as data type are already filled up for all intended fields. You may also notice a icon of 'Key' appearing before the 'EmpID' field indicating it to be the Primary Key of the table.

For the benefit of laterday referencing, you may choose to record a brief description of different attributes in the column 'Description'. Filling up of description of fields is optional, and has no bearing on the design of table. Even then it is desirable, especially if your database is likely to be used by other programers as well. Besides the data type, there are several other properties that can be set independently for each field. Appearing as 'Field Properties' at bottom part of the design

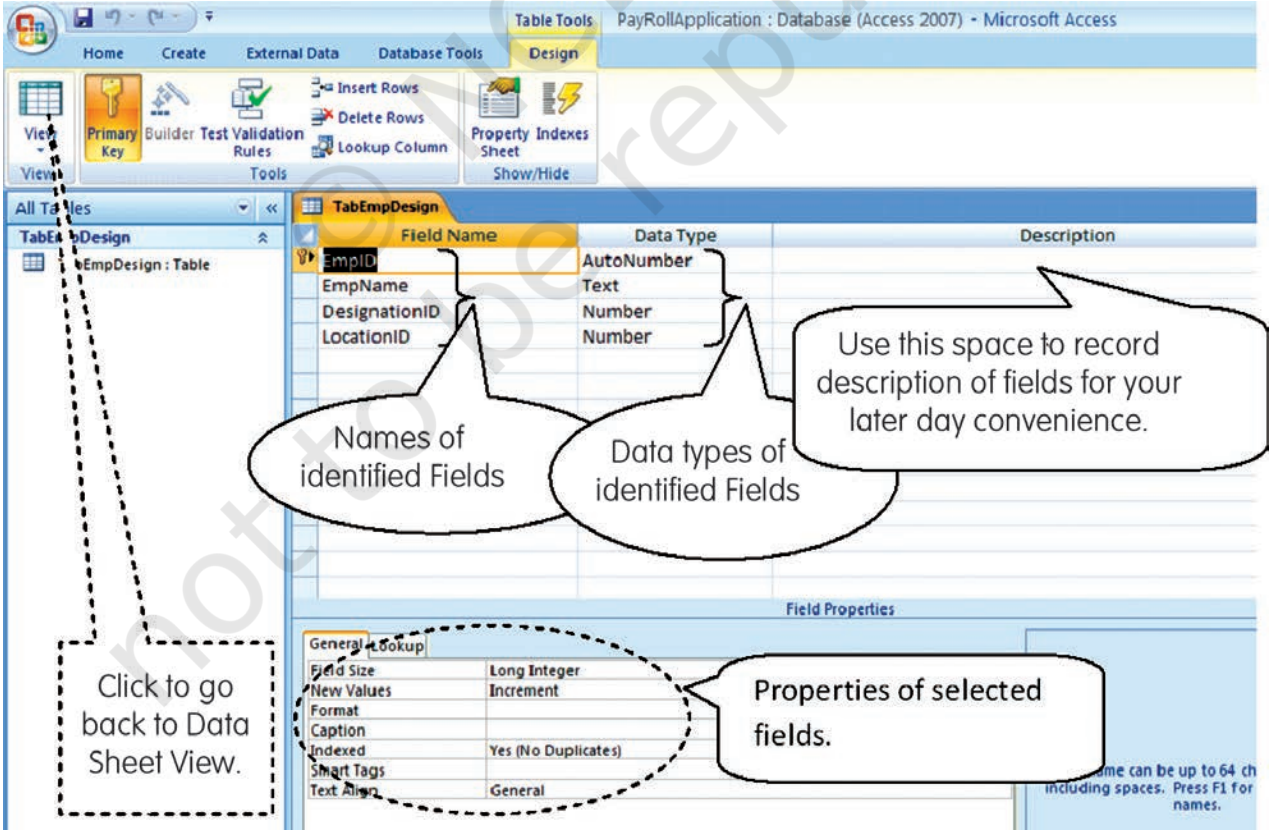


Figure 5.9 : Design View of the Table

view, these parameters of a field will vary with the choice of data type. A narration of each property (when you click the desired property) also appears in a hint box on right hand side, which makes understanding of field properties quiet simple. However, we will stick to default properties of all fields, and ignore this area altogether.

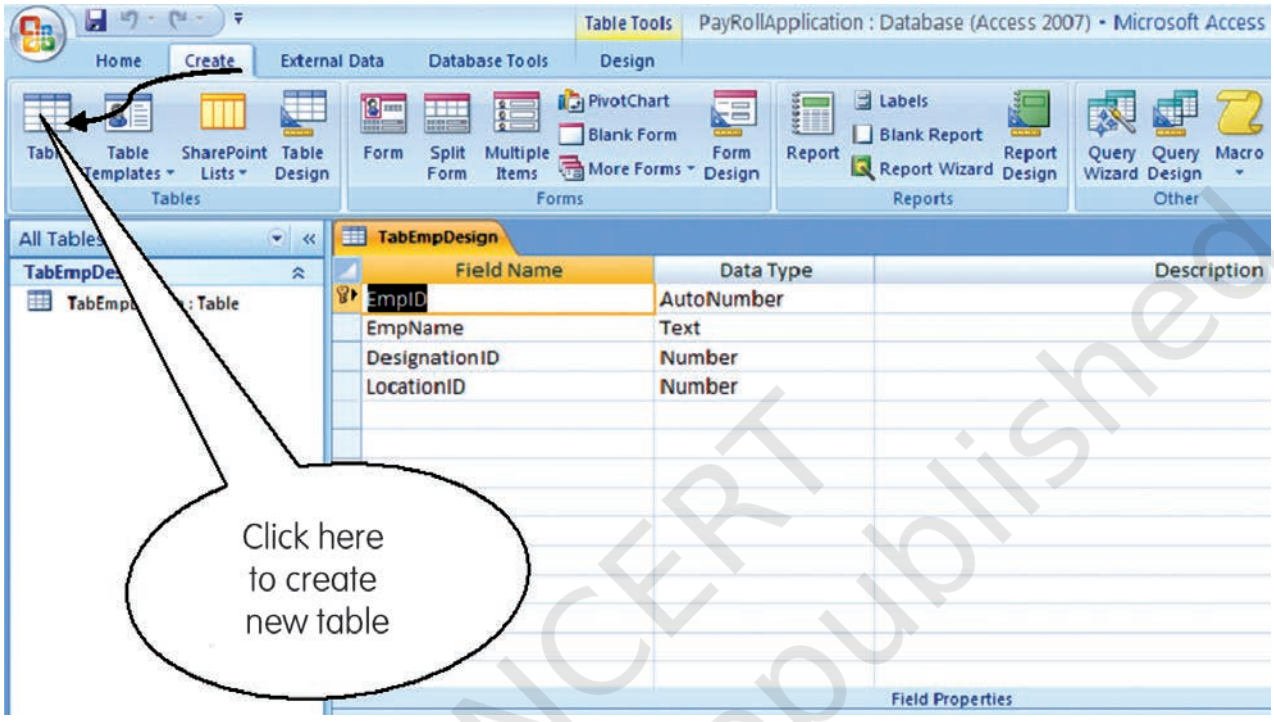


Figure 5.10 : Creating New Table

Having fully completed the structure and data of our first table and also having understood the various conccepts related to its design, we can now move on and create rest of the four tables. Click on **Create Tab > Table Button** as illustrated in Figure – 10. The Data Sheet view of the default table (Table1) will appear again. You can choose this view or the Design View to create the rest of the tables and save them under intended names. For your conveniene, the field names and data types of all five tables are summarised as under:

Sl. No.	Field Name	Data Type	Description
1.	**TabEmpID**		
	EmpID	AutoNumber	Unique employee ID – Primary Key.
	EmpName	Text	
	DesgID	Number	
	LocationID	Number	
2.	**TabDesignations**		
	DesgID	AutoNumber	Unique Designation ID – Primary Key.

	Designation	Text	
	TA	Number	Transportation Allowance – fixed as per designation.
3.	**TabLocations**		
	LocationID	AutoNumber	Unique Location ID – Primary Key.
	Location	Text	
	RateOfHRA	Number	% Rate of House Reent Allowance.
4.	**TabDARates**		
	MonthID	AutoNumber	Unique MonthID – Primary Key.
	SalMonth	Text	With values as 'Jan-2007', "Feb-2007" etc.
	RateOfDA	Number	% Rate of Deanress Allowance.
5.	**TabMonthly Salary**		
	SalaryID	AutoNumer	Unique SalaryID – Primary Key.
	MonthID	Number	
	EmpID	Number	
	Basic	Number	Basic pay based on level & attendance of employee.
	DedForPF	Number	Deduction for PF opted by employee & as per rules.

Having completed the designs of all data tables, we will now move on to the task of establishing relationships between different tables. Click on the ***Databse Tools*** Tab and then ***Relationships*** button under ***Show/***

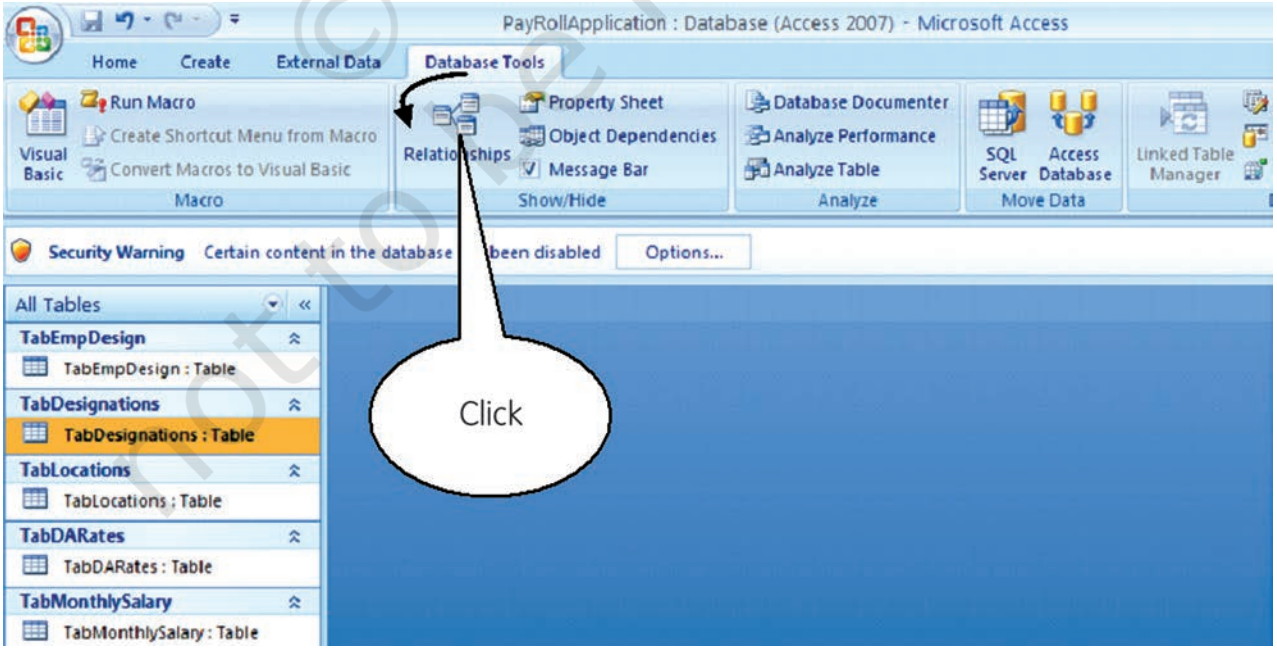


Figure 5.11 : Creating Relationship between Table

Hide group as shown in Figure - 5.11. In the working area, a ***Show Table*** dialogue box will appear as shown in Figure - 6.12. In case you do not see this dialogue box click on ***Design > Show Table***. In the ***Show Table*** dialogue box, select a table and clik ***Add*** button to add it in the relationship window. Add all the five tables in this manner. Close the Show Table dialogue box by clicking on ***Close*** button.

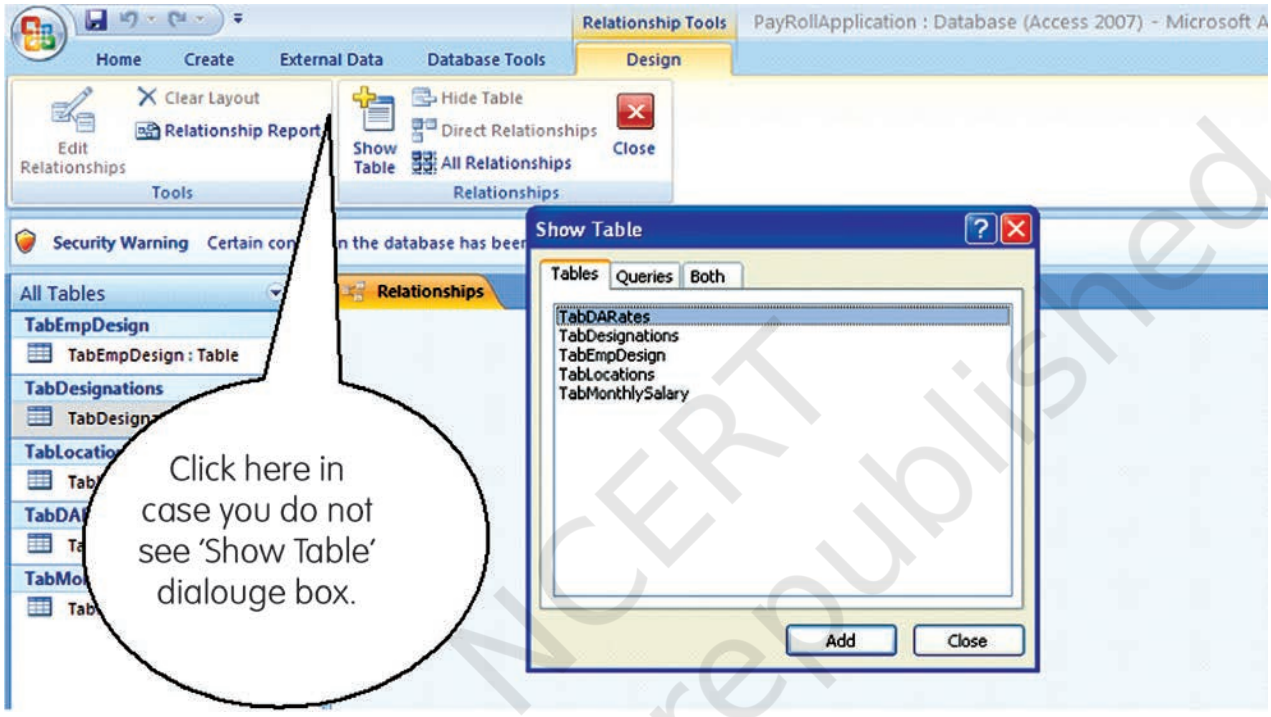


Figure 5.12 : Adding of Tables for Establishing relationships between them

In the working area you will see all five table objects, each detailing the fields within them. You can reposition these table objects any where within the relationship window. To do this, point the mouse pointer in the caption area of the selected table object, hold down the left mouse button and then drag the mouse to shift table to its new location as shown in Figure -5.13. Now to create a relationship between TabMonthlySalary and TabDARates, position the mouse pointer over MonthID in the TabMonthlySalary table object, hold down the left mouse button, drag the pointer right to MonthID in the TabDARates, and then release the mouse button. A ***Edit Relationships*** dialogue box will appear as soon as you release the mouse button. You will notice that the dialogue box shows relationship type as **One-To-Many** (see Figure-5.13).

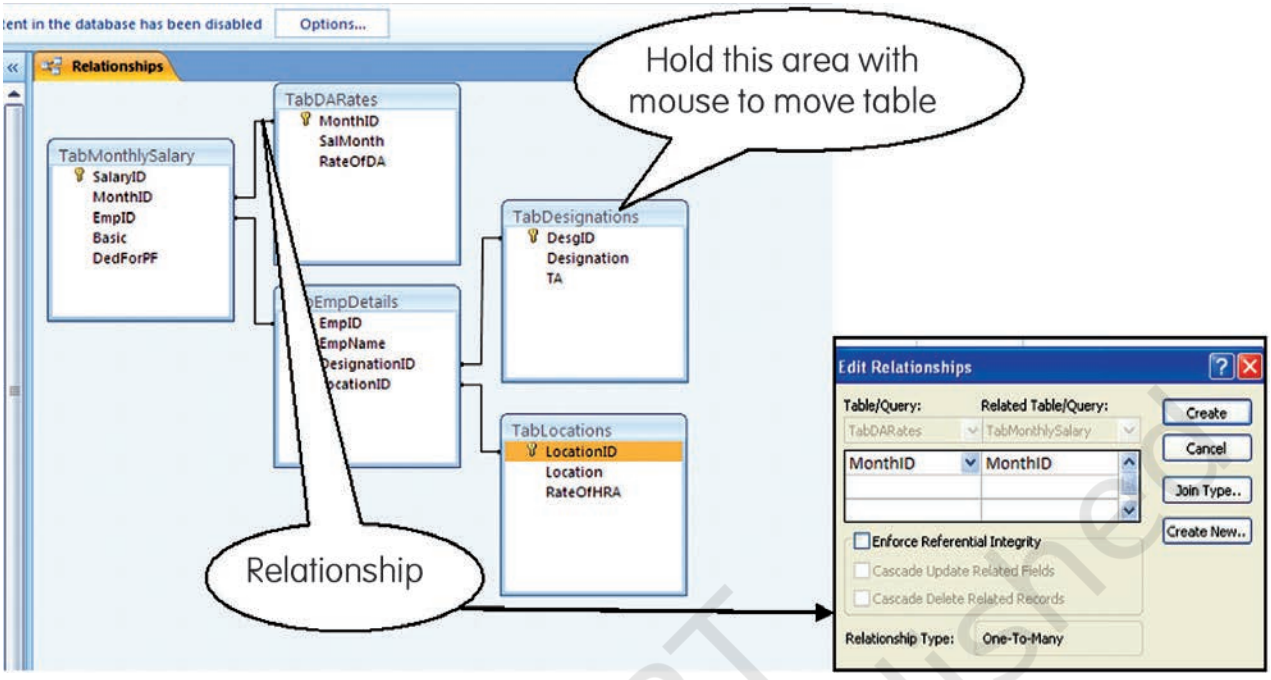


Figure 5.13 : Relationship between different Tables

Access has determined the relationship type based on the fields selected for joining the two tables. Click on the ***Create*** button. A black line will appear joining the two tables at the common field. This line, also referred to as **Join Line**, establishes the relationship between TabMonthlySalary and TabDARates using the common field of MonthID. Remember, MonthID forms Primary Key in TabDARates and Foreign Key in TabMonthlySalary and hence is the relationship One-To-Many type. You may now repeat above steps to establish other relationship between the tables as indicated in Figure - 5.13. The overall relationship between different tables are summarised as under:

Table	Related Table	Common Field	Relationship Type
TabDARates	TabMonthlySalary	MonthID	One-To-Many
TabEmpDetails	TabMonthlySalary	EmpID	One-To-Many
TabDesignations	TabEmpDetails	DesgnID	One-To-Many
TabLocations	TabEmpDetails	LocationID	One-To-Many

5.5 CREATION OF QUERY IN MICROSOFT ACCESS

As discussed in Section 4.0, we will now get on to the task of handling computational fields of 'DA', 'HRA', 'TA', 'GrossSalary' and 'NetSalary' using **Queries**. The Queries provide the real power to a database in terms of its capabilities to answer more complex requests (or **queries)**. In case of Access, Queries provide the capability of combining data from multiple tables and placing specific conditions for the

retrieval of data. In its simplest form, you may consider a Query to be another tabular view of your data showing information from one or more tables.

Click on **Create > Query Design**. A ***Show Table*** dialogue box will appear with a Query Table in the background as shown in Figure – 5.14. In case you do not see this dialogue box click on ***Design > Show Table***. In the ***Show Table*** dialogue box, select a table and clik ***Add*** button to add it in the relationship window. Add all the five tables in this manner. Close the Show Table dialogue box by clicking on ***Close*** button. In working area above the Query Table, you will see all five table objects (with complete list of their fields) along with the one-to-one relationship that has been established between tables earlier. You can reposition these table objects in the manner discussed earlier. In the portion below Table objects, you will see blank columns that represent columns in the Query results datasheet. These columns (see Figure – 5.14) are also referred to as the ***Design Grid***.

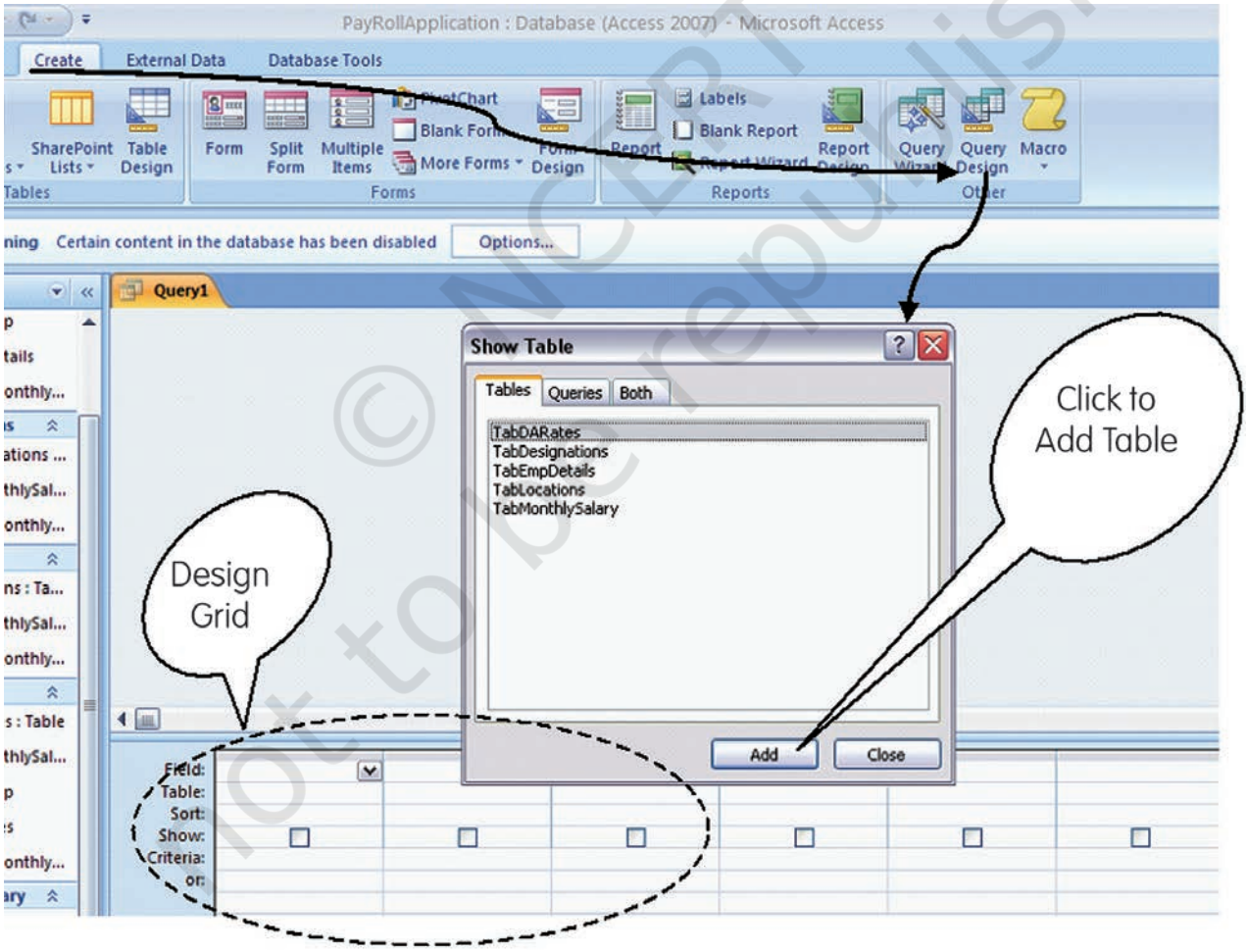


Figure 5.14 : Creation of Query

In the first column of the design grid, click on the Table row. A dropdown button will appear on this cell. Click on the button to get the dropdown list of tables and click on 'TabMonthlySal'. Now in the same column, click on Field row and get the dropdown list of all fields in 'TabMonthlySal'. Click on the 'SalaryID' field. In this manner the SalaryID (from TabMonthlySal) has been selected for view in the Query. The above operation could have also been done simply by double clicking the 'SalaryID' in the list box of TabMonthlySal object. You could have also done this by positioning the mouse pointer on the identified field of a table object, held down the left mouse button, and dragged it to the field row of the desired column of the design grid, and then released the mouse button.

You shall take care to fill different fields from Table object into the Design Grid in the same order in which you want to display these fields in the Query Results data sheet. Now select any of the described methods to fill other columns of the design grid in the specified order with fields of Month ID (TabMonthlySalary), SalMonth (TabDARates), EmpName (TabEmpDetails), and Basic (TabMonthlySalary) as illustrated in Figure - 5.15. In case you may not require the need of displaying MonthID, you may click on this field at the row marked as 'Show' to uncheck the show-check-box. At this stage, if you click on the ***Run*** button under ***Results*** group of ***Design*** Tab, you will see the result of Query displayed with the fields of SalaryID, SalMonth, EmpName, and Basic. Clicking of 'Run' actually tells the Access to

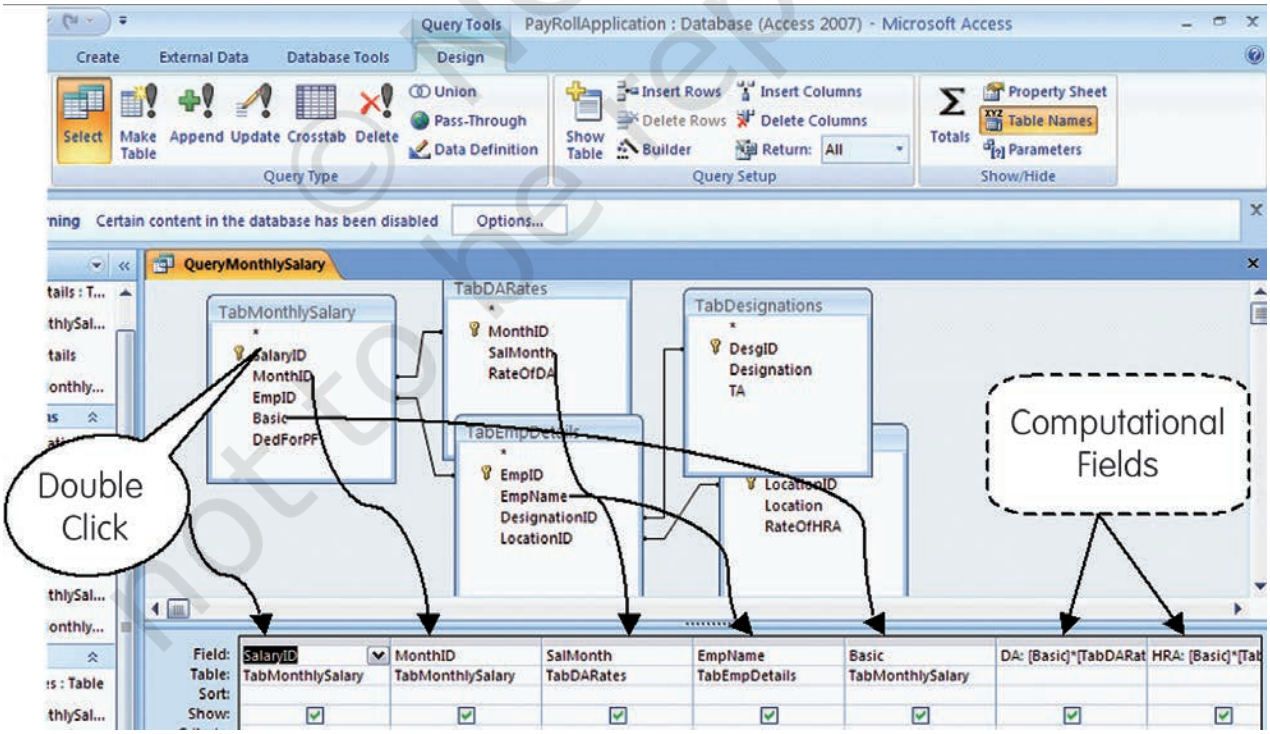


Figure 5.15 : Creation of different Fields in the Query Table

execute all instructions stored by you in the Query, and to display the results. You may adjust the column widths by any of the methods described earlier for Tables. Click on the ***Save*** button on the Quick Access toolbar. At the Save As disalogue box, type QueryMonthlySalary and click ***OK*** to save our Query.

Evidently our Query is not complete as we are not seeing the computational fields and the field of DedForPF in the ***DataSheet View***. Go back to the Design Grid view by clicking on Design View under Home Tab. In the Design Grid, Click on the Field row of the first balnk column (after Basic) and type '**DA: [Basic]*[TabDARates.RateOfDA]/100**'. In this expression, the word 'DA' coming before the colun(:) will be treated as the name of the computational field which will appear as heading of the column. The rest of the typed part will form the actual expression which will be evaluated and its value displayed on clicking of **Run** command. You may notice that in our expression for 'DA', we have made use of 'Basic' field which is already forming a part of Query by taking it within the square parenthesis ([ ]). We have also made use of 'RateOFDA' field which is not appearing in the Query columns; and for this we recorded the corresponding Table name as well as Field name separated by a period (.) within the square parenthesis. It may be easily comprehendable that the operator '*' and '/' stand for multiplication and division respectively. At this stage if you click on the '**Run**' command, you will notice the last field for 'DA' with its computed values for different employees as per the % DA Rates for different months.

Come back to the design view (by Clicking ***Design View*** button under ***Views*** Group) and repeat above listed procedure to fill up the next field of Query for computing HRA using the expression '**HRA: [Basic]*[TabLocations.RateOfHRA]/100**'. The next column in the Query is for the **TA** field, which is to be included as such from the Table '**TabDesignations'** using any of the methods enumerated earlier. Next, create the field for 'Gross Salary' by typing the expression '**GrossSalary: [Basic]+[DA]+[HRA]+[TA]**'. Now we can also include the field of '**DedForPF**' from the '**TabMonthlySalary**'. Finally, we shall include the field of NetSalary using the simple expression '**NetSalary: [GrossSalary]-[DedForPF]**'. At this stage, save your Query as instructed earlier and also run the '**Run**' command to look back the outcome of your effort in the Data Sheet View.

While Figure -5.15 gives an illustration of the Design View of our Query, the Figure - 5.16 shows the Datasheet View of the Query. Your view may be little different based on the actual data that you have entered in different tables. As shown in Figure 6.16, you may notice that the total dataset of the Query is composed of all data rows corresponding to the Table 'TabMonthlySalary'. You may restrict the dataset to any restricted values for any of the specified field. For example, we may want only the data corresponding to November 2007,

for which the MonthID is '6'. With this objective, click on the **Criteria** row under **MonthID** field and type '=6' as shown in Figure - 5.17. Now if you click on '**Run**' command, you will see the data records corresponding only to the month (SalMonth) of Nov 2007'.

QueryMonthlySalary

SalaryID	SalMonth	EmpName	Basic	DA	HRA	TA	GrossSalary	DedForPF	NetSalary
1	Nov. 2007	Ram Kishore	25000	6500	7500	7000	46000	5000	41
2	Nov. 2007	Kishan Sharma	22000	5720	4400	5000	37120	3000	34
3	Nov. 2007	Rupali Varma	20000	5200	3000	3500	31700	2000	29
4	Nov. 2007	Surjeet Singh	16000	4160	2400	3500	26060	2000	24
5	Dec. 2007	Ram Kishore	25000	6500	7500	7000	46000	7000	39
6	Dec. 2007	Kishan Sharma	22000	5720	4400	5000	37120	3000	34
7	Dec. 2007	Harish Bajaj	5500	1430	1650	1000	9580	1000	8
8	Dec. 2007	Indira Jain	17000	4420	2550	3500	27470	2000	25
9	Jan. 2008	Ram Kishore	25000	7500	7500	7000	47000	6000	41
10	Jan. 2008	Kishan Sharma	22000	6600	4400	5000	38000	2000	36
11	Jan. 2008	Susan Jacob	17000	5100	5100	2500	29700	2500	27
12	Feb. 2008	Ram Kishore	25000	7500	7500	7000	47000	5000	42
13	Feb. 2008	Rupali Varma	20000	6000	3000	3500	32500	4000	28
14	Feb. 2008	Surjeet Singh	16000	4800	2400	3500	26700	3000	23
15	Feb. 2008	Susan Jacob	17000	5100	5100	2500	29700	1000	28
16	Feb. 2008	Dharam Singh	11000	3300	2200	1000	17500	1500	16
17	Mar. 2008	Kishan Sharma	22000	6600	4400	5000	38000	3000	35
18	Mar. 2008	Rupali Varma	20000	6000	3000	3500	32500	2500	30
19	Mar. 2008	Susan Jacob	17000	5100	5100	2500	29700	3000	26
20	Apr. 2008	Ram Kishore	25000	9250	7500	7000	48750	5000	43
21	Apr. 2008	Kishan Sharma	22000	8140	4400	5000	39540	3000	36
22	Apr. 2008	Surjeet Singh	16000	5920	2400	3500	27820	3500	24

Figure 5.16 : Datasheet View of the Query

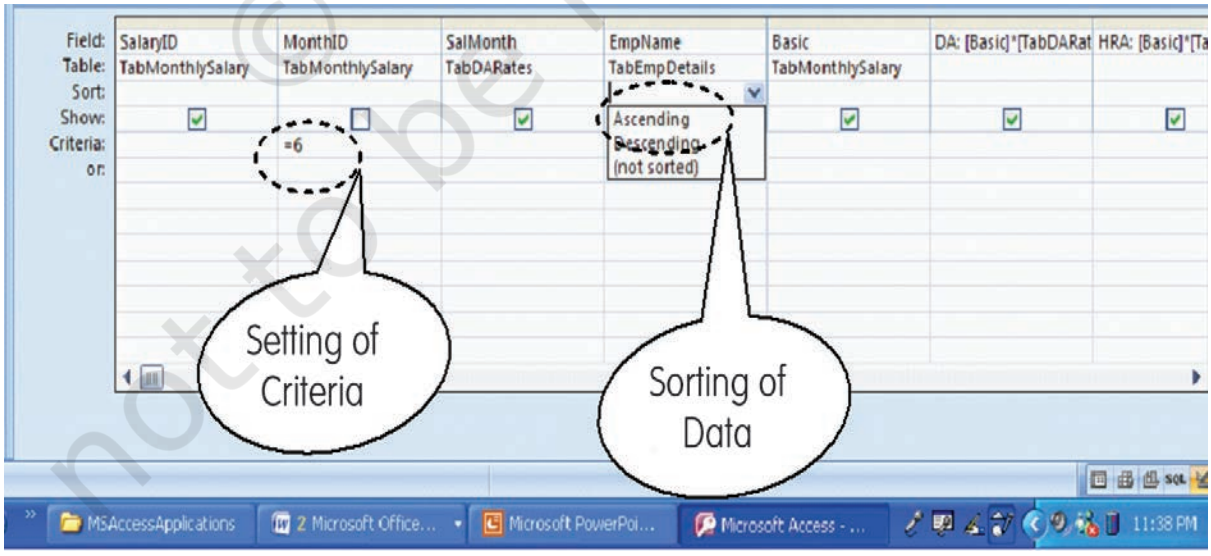


Figure 5.17 : Setting of Criteria and Sorting of Data in a Query

For a better presentation, you may also like to sort the records in ascending or descending order against a chosen field of the Query. Thus, we may like to arrange the records of Nov.2007 as per alphabatically ordered names. For this, click on the **Sort** row below the EmpName field. Now Click on the pull down button that appears on the end of the cell and select ***Ascending*** (see Figure – 5.17). Run the Query and notice that the records are alphabatically arranged as per names of the employees.

5.6 CREATION OF FORMS IN MICROSOFT ACCESS

So far, we have learned the concepts behind organising the information in a database using Tables and retrieving such information using Queries. Now we will look into the mechanisms for placing information into the tables as and when the need arises. The Microsoft Access provides two primary mechanisms to achieve this goal. The first method, which we are already familiar with, is to simply bring up the table in a window by double-clicking on it and adding information to the new blank row at the bottom. We can also make changes in any other row as one would do in case of a spreadsheet. You can save the changes in the table by clicking on the ***Save*** button on the Quick Access toolbar.

Box - 5.12

The Access 2007 provides facility for creation of a **Simple Form** *wherein you can enter information for one record at a time. Access also provides tools for creation of a* **Split Form** *which shows the underlying datasheet in one half of the section and a Form in other half for entering information in the record selected in the datasheet. The two views in this form are synchronised so that scrolling in one view causes the scrolling of other view to same location of the record.*

As second mechanism for information handling (i.e. for addition, modification and deletion), the Access provides a user-friendly ***Forms*** interface that allows users to enter information in a graphical form, and this information transparently passes to the underlying database. This method is more user-friendly for the data entry operator, though it may entail a little more work on the part of the database designer.

For our Pays Roll Application, let us create a simple Form for the Table TabMonthlySalary – which is one table likely to be used more frequently by the bill clerk. In the Navigation Pane of **PayRollApplication,** Click once on the **TabMonthlySalary.** Having selected the intended Table thus, Click on ***Create*** Tab and then on ***Form*** under Forms Group. The Form View of the TabMonthlySalary will appear in the working area of Window showing first record of the Table as illustrated in Figure 18. At the bottom of the Form you will see Navigation bar with buttons for 'First Record', 'Previous Record',

'Next Record', 'Last Record' and 'New Record'. In the middle of Navigation bar, you will also see the number of current record (e.g. 1 of 24). Click on the buttons for Next Record a few times to scroll through a few records in Form View. You may also click other buttons to understand their functions. If you click on New Record button, you will notice that a new record will get displayed with '***SalaryID'*** displaying **(New)** while all other fields remaining blank. You may recall that **SalaryID** is the **Primary Key** of this table with **AutoNumber** as its data type.

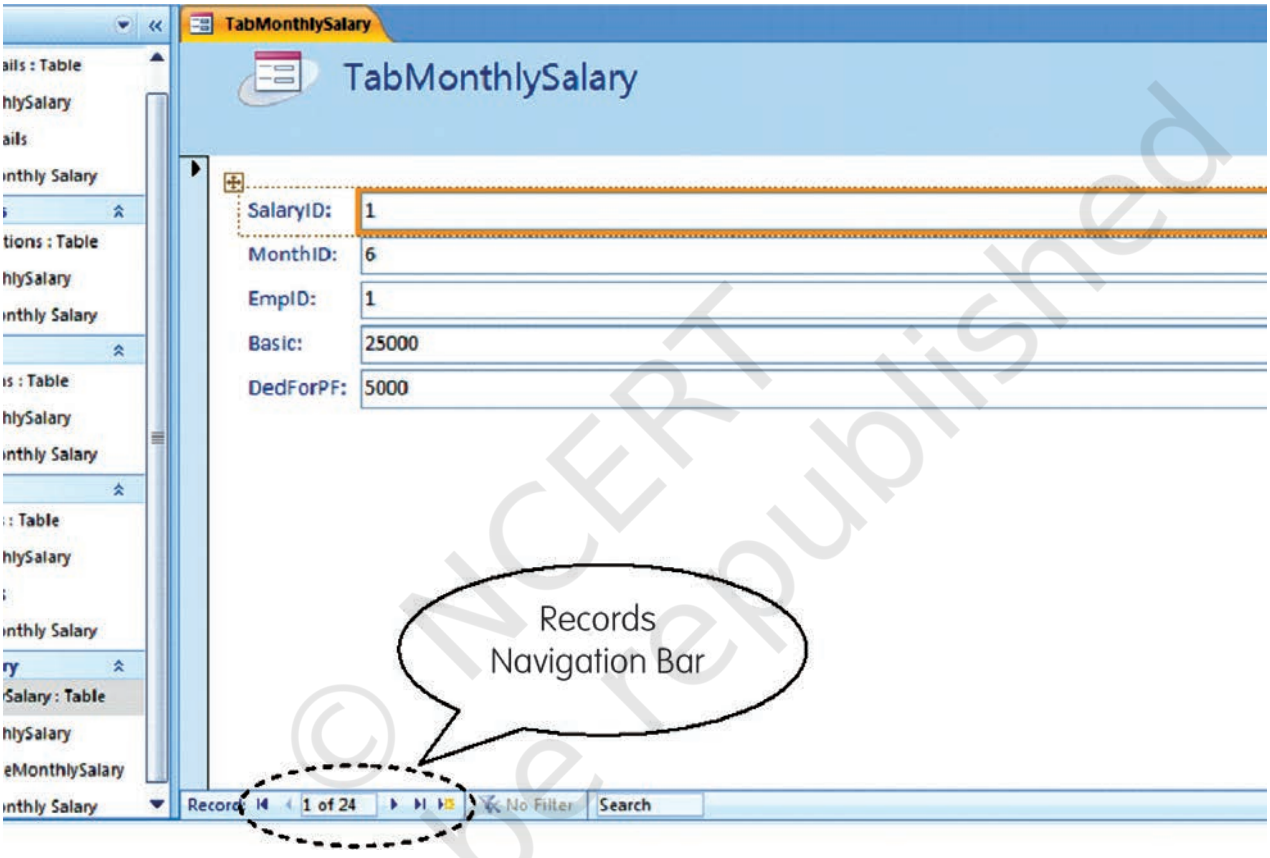


Figure 5.18 : Creation of Forms

We can make some changes in the size of data boxes and their location within the Form. With this objective, let us first change the view of Form to Design View by clicking on ***Home > Design View***. In the Design View, click on any of the Text Boxes. Move the mouse pointer to right end of the highlighted boundary of the Box till a Left-Right arrow appears. Press the Left Mouse button, and holding it there move the mouse towards left to reduce the width of Text Box to desired size. You will notice that the sizes of all Text Boxes have reduced automatically. Now select all text boxes by clicking once on each box while pressing the Shift key. Now move the Mouse pointer over any of the box till a sign with four arrows (similar to a + sign) appears. Press the Left Mouse button, and holding it there move the mouse towards right to shift the Text Boxes to a more central location.

We know that in TabMonthlySalary, the field of MonthID is a Foreign Key. In order to have meaningful values in our final Query (i.e. QueryMonthlySalary), this field shall contain only those values that are already stored in TabDARates. We can impose this condition in the Form while we are in Design View. Click and select the MonthID Text Box. Keeping the mouse pointer within this Box, click the right mouse button and select ***Change To > Combo Box*** as illustrated in Figure –19. You will notice that the Text Box of MonthID field will change to a Combo Box with a pull down button appended at the end of box. With the ***MonthID*** ComboBox still selected, click on the ***Property Sheet*** Button under **Tools** Group. This will open up the Property Sheet of the **MonthID** Combo Box as shown in Figure - 6.20. In the ***Data*** Tab of Property Sheet (which may be already open before you), click on the ***combo*** button against '***Row Source***' property to see the pull down list of Tables; and select '**TabDARates**'. Note that you will be able to see the Combo option under 'Row Source' property, only if, the '**Row Source Type**' property is selected as '**Table/Query**'. Ensure that the '**Bound Column**' property has a value of '1', which corresponds to the First Column – i.e. '**MonthID**' – of the '**TabDARates**'. Having thus reset the property of MonthID field of the form, change the view to **Form View**. You will notice that the Form now shows a Combo Box against the **MonthID** field and in its pull down list only those MonthIDs will appear which are already stored in the **TabDARates**. You may realise that in **TabMonthlySalary**, the field of **EmpID** also needs to be given similar treatment; and its values shall be restricted to the values of '**EmpID**' given in the **TabEmpDetails**.

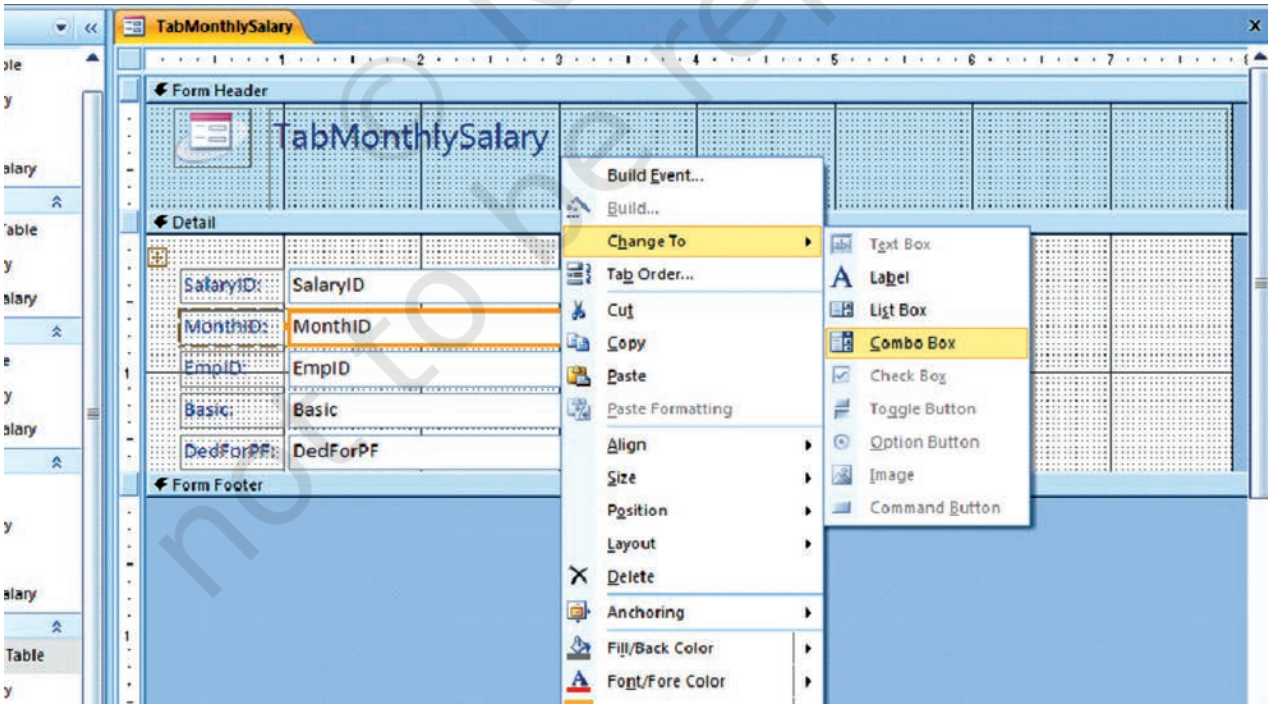


Figure 5.19 : Changing of Textbox to a Combo Box in the Form Design View

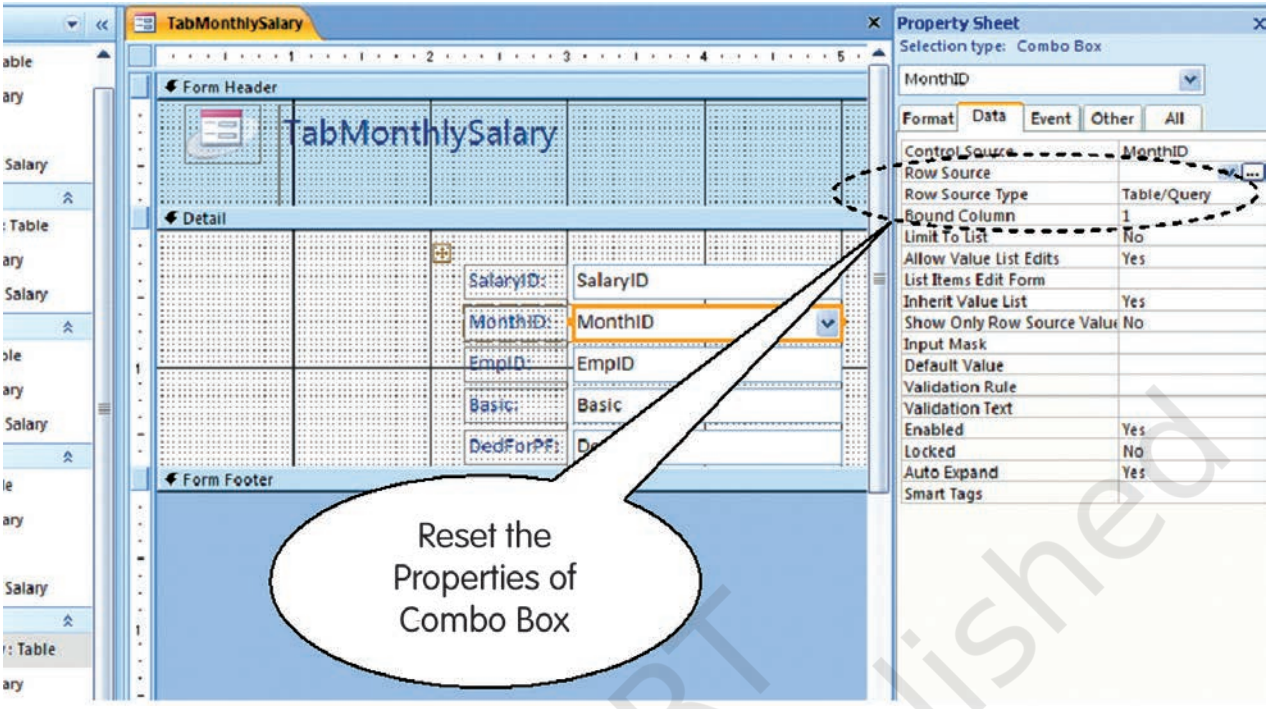


Figure 5.20 : Setting the Properties of Combo Box

Many more changes can possibly be done in our Form to make it more user-friendly. However, these may entail the use of 'Macro Builder', 'Query/Expression Builder' and 'Code Builder'. These topics are beyond the scope this book, though you may explore them gradually after developing familiarity with presently covered areas of Access programming. You may also develop suitable designs of Forms for other Tables of the Payroll application.

5.7 CREATION OF REPORTS IN MICROSOFT ACCESS

Information stored in a Table or Query can be easily printed while it is in Datasheet view. This can be done by using ***Print*** command given under Office Button on left hand top corner of the window. Even a Form view can be directly printed in similar manner. A **Report** is also used for the print purpose, though it allows for more flexibility in selecting the fields to print, and to have more control on the overall layout and format of the print output. The **Report** in Access is thus another object which is designed to print information from the database on to the screen, or to a file or directly to the printer.

Here also you can take the elaborate route of step-by-step design as was done in case of Tables, Query or Form. However in this case, you may find the ***Report Wizard*** becoming a more favorable tool as it guides the designer through a series of dialogue boxes to create the most suitable Report.

Evidently in our case, we will be interested in generating the Report for the '**QueryMonthlySalary**'. With this intent, Click on 'QueryMonthlySalary' in the Navigation Pane, then click on ***Create*** Tab, and then on ***Report Wizard*** under Reports Group. In the dialogue box that appears with QueryMonthlySalary Query already selected, move all fields from the ***Available Fields*** List Box to the *Selected Fields* List Box using appropriate ***Arrow*** button as shown in Figure - 5.21. Click ***Next*** to see the second dialogue box asking for Grouping Levels. Here select 'SalMonth' and click on the Arrow button (else double click the 'SalMonth') to print data in groups of different salary months (See Figure – 5.21). Click ***Next*** to reach the third dialogue box asking for sorting order of the records. Here you can click on the **Pull-down Arrow** next to the first text box and then select '**SalaryID**' from the drop down list so as to arrange the records in ascending order of the salary ID within each data group - as illustrated in Figure - 5.22. Click Next to see the dialogue box for layout of the Report. Retain the default '**Stepped'** layout but change the orientation of Report to '**Landscape'** so as to see all the columns of the Report in a single page view (see Figure - 5.22). Again Click ***Next*** to see the dialogue box for selecting style of Report. You may choose any style and later on experiment with the others as well. One more click of ***Next*** will take you to the dialogue box seeking the name of Report object. Type **ReportMonthlySalary** and click on the ***Finish*** button to see the Report appearing in working area of window in its **Print Preview** form.

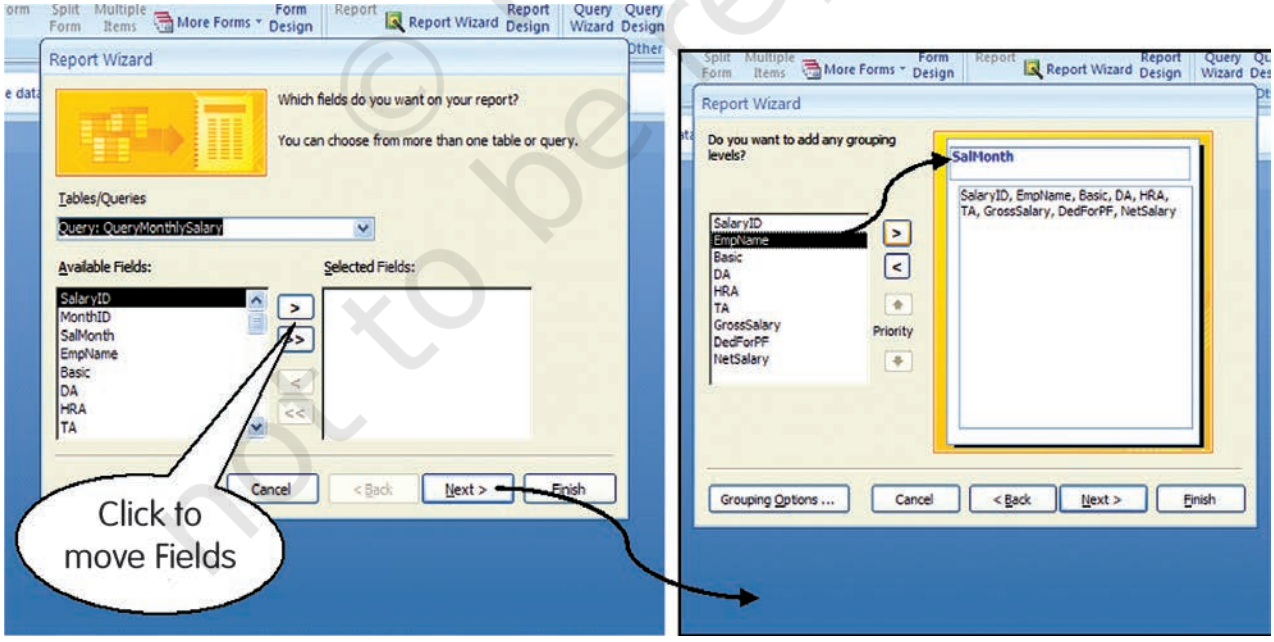


Figure 5.21 : Report Wizard Dialogue Boxes

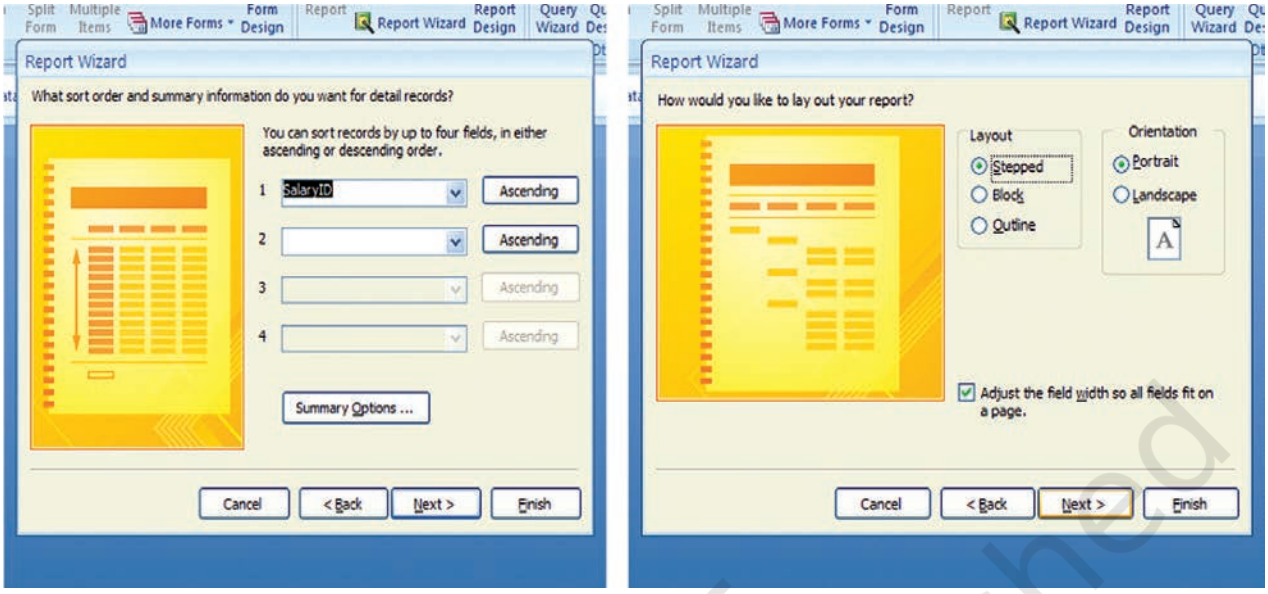


Figure 5.22 : Report Wizard Dialogue Boxes

You may notice that the extent of data in above designed Report is limited as per the **Criteria** laid down in our Query '**QueryMonthlySalary**'. By changing the Query suitably, we can modify the Report for generating monthly as well as annual Pay Rolls. We can also change the Criteria of our Query to get individual Report for each month (i.e. Pay Slip) or for the whole of year (i.e. Annual Statement).

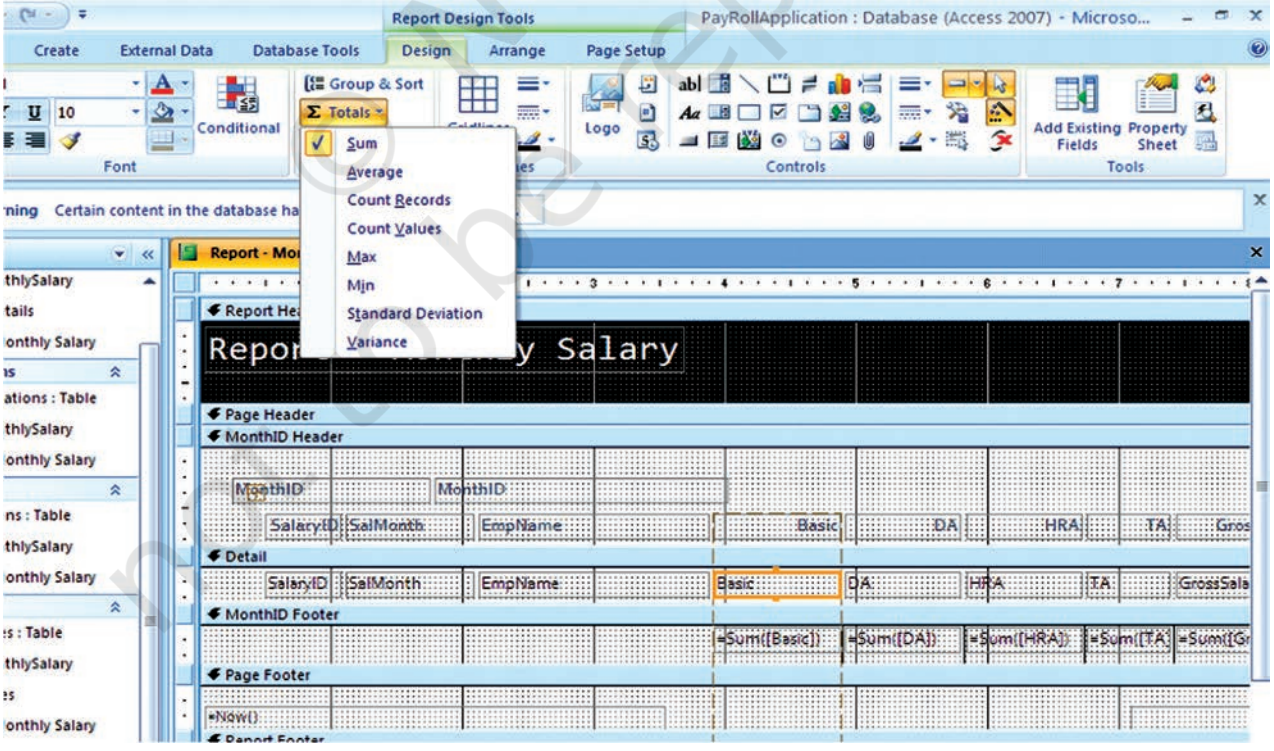
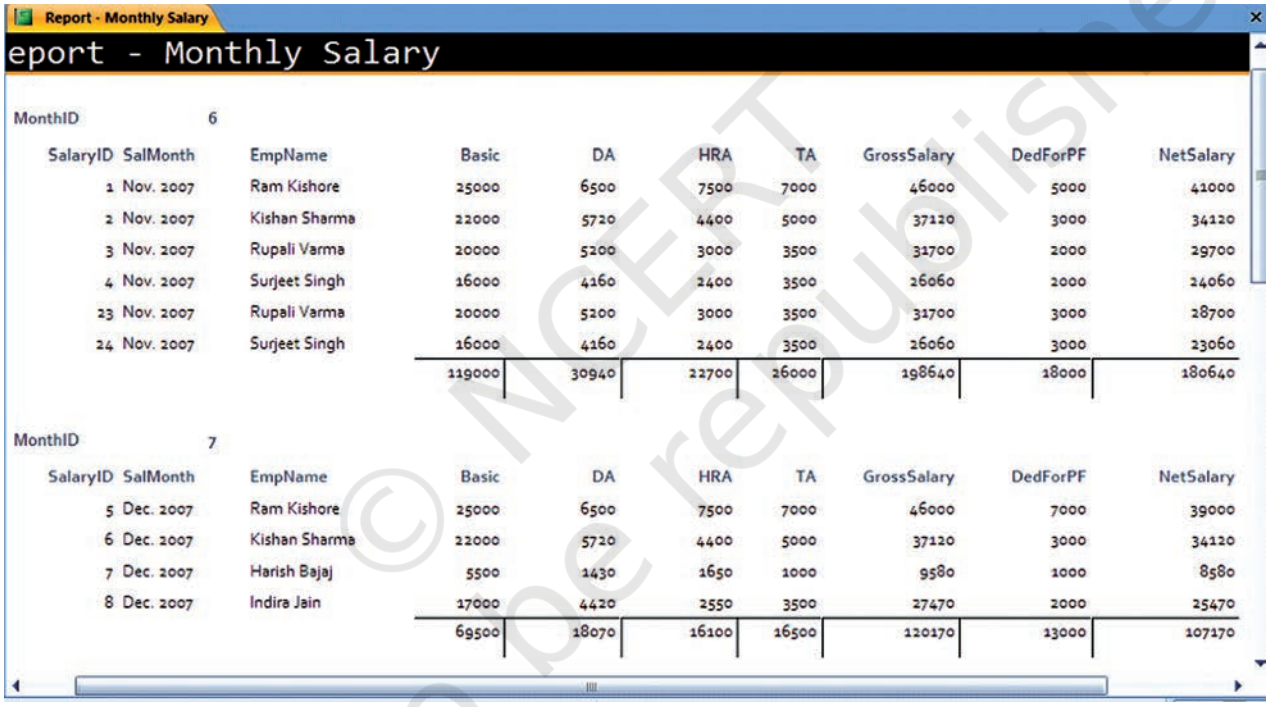


Figure 5.23 : Adding of Sum Expression in the Report

We may like to improve the Report further by getting sum total of all the values under different columns for each group. For example, we may want to see the total Basic of all employees for November 2007 group. For such modification, open the Report in Design View by Clicking on ***Home*** Tab and then clicking on ***Design View*** under Views Group. In the design view as shown in Figure - 5.23, select the field **Basic** by clicking it once. Now click on ***Totals*** under '**Grouping and Totals**' Group of ***Design*** Tab. In the pull down menu that appears, click on '**Sum**'. You will notice that an expression **'=Sum([Basic])**' will appear below the Basic Text Box. You can repeat above steps with other fields containing numerical (both stored as well as computed) values. Now if you click on **View** button (i.e. click ***Design > Report View***), you will see the sum total of each column (with in each group) appearing in the report as illustrated in Figure 5.24.


Report - Monthly Salary

eport - Monthly Salary

MonthID 6

SalaryID	SalMonth	EmpName	Basic	DA	HRA	TA	GrossSalary	DedForPF	NetSalary
1	Nov. 2007	Ram Kishore	25000	6500	7500	7000	46000	5000	41000
2	Nov. 2007	Kishan Sharma	22000	5720	4400	5000	37120	3000	34120
3	Nov. 2007	Rupali Varma	20000	5200	3000	3500	31700	2000	29700
4	Nov. 2007	Surjeet Singh	16000	4160	2400	3500	26060	2000	24060
23	Nov. 2007	Rupali Varma	20000	5200	3000	3500	31700	3000	28700
24	Nov. 2007	Surjeet Singh	16000	4160	2400	3500	26060	3000	23060
			119000	30940	22700	26000	198640	18000	180640

MonthID 7

SalaryID	SalMonth	EmpName	Basic	DA	HRA	TA	GrossSalary	DedForPF	NetSalary
5	Dec. 2007	Ram Kishore	25000	6500	7500	7000	46000	7000	39000
6	Dec. 2007	Kishan Sharma	22000	5720	4400	5000	37120	3000	34120
7	Dec. 2007	Harish Bajaj	5500	1430	1650	1000	9580	1000	8580
8	Dec. 2007	Indira Jain	17000	4420	2550	3500	27470	2000	25470
			69500	18070	16100	16500	120170	13000	107170

Figure 5.24 : Print Preview of Monthly Salary Report

At this stage, close the Report by clicking on the **Cross** button immediately above the report. Save the application once again by clicking on the ***Save*** button under **Quick Access Bar**. Our PayRoll Application in Access has been completed and saved. You may now close your application by clicking on the ***Close*** button under '***Office Button***' or by simply clicking on the **Cross** button at right-hand top corner of the window.

Summary

- Database Management System (DBMS) provides a variety of software tools for organising, processing and querying data in a flexible manner. MS-Access, Oracle, SQL Server, IBM-DB2 are examples of DBMS software.
- In DBMS, data is organised in tables (similar to a file). A table has a number of rows (or records) and columns (or fields or attributes). Each row contains a record of information, for example of an account head or a party or a transaction as per the need. The information in a row consists of a sequence of columns or attributes, such as transaction number, transaction date, etc, or it could be party's name, party's address, etc.
- One of the tasks in analysis of requirement is to identify and list out the information required including its elements. These elements of information become columns (attributes) in appropriate tables.
- Data (set of attributes) should be logically structured so as to put them in various tables. The goal of such structuring is to reduce data redundancy, to achieve data consistency as well as to enhance efficiency for adding, updating and querying operations on database.
- Normalisation is the process for removing data redundancy.
- Since the data stored in different tables may be related, such relationship is implemented by establishing links between tables. The database created on the basis of such relationships between different tables is called relational database.
- Relationship between tables is established with the help of primary key and foreign key. Primary key consists of minimum possible one or more than one attributes of a table, which uniquely identifies each row of that table. Foreign key consists of set of attributes, which from primary key in another (related) table.
- Most of Computerised Accounting Systems are multi-user systems. These systems use 'server database' unlike single-user (or desktop) systems using 'desktop database'. In a multi user system, a user interacts with the software though the user interface, which is also termed as 'front-end'. Database, which is kept on a server, is termed as a 'back-end'.
- MS-Access is an example of 'desktop database'. Oracle, SQL Server, IBM-DB2 is examples of 'server databases', Desktop databases may be satisfactory for SOHO (Small Office Home Office) organisations as they offer inexpensive and simple solutions to many of business data storage and processing requirements.
- In order to provide security and consistency of data, database is not directly accessible to users. Any addition or retrieval of information from database is done by user-friendly programs. Database is thus rightly referred to as 'back-end' while the interactive program, that includes user interface, is termed as 'front-end' of a database application.

EXERCISE

Q1. Multiple Choice Questions

1. 'DBMS' stands for:
   a. Drawing Board Management Software
   b. Dividend Based Marking System
   c. Data Base Management System
   d. Data Base Marking Software.
2. MS Access is a:
   a. Word processing Software
   b. Presentation Software
   c. Spread sheet Software
   d. Data Base Management Software.
3. The term 'field' as applied to database table means:
   a. Vertical column of the table
   b. Size of the table
   c. Horizontal row of the table
   d. Name of the table.
4. The term 'record' as applied to a database table mean:
   a. Vertical column of the table
   b. Size of the table
   c. Horizontal row of the table
   d. Name of the table.
5. The common fields used in a relationship between tables are called:
   a. Joint fields
   b. Main fields
   c. Key fields
   d. Table fields.
6. The existence of data in a Primary key field is:
   a. Not necessarily required
   b. Required but need not be unique
   c. Required and must be unique
   d. All of above

7. The existence of data in a secondary key field is :
   a. Not necessarily required
   b. Required but need not be unique
   c. Required and must be unique
   d. All of above
8. SQL stands for:
   a. Simple Questions Language
   b. Simple Que line up
   c. Singular Quantity Loading
   d. Structured Que Language
9. The default extension of MS Access (2007) file is :
   a. .accbd
   b. .exl
   c. .doc
   d. .exe
10. Wizards in MS Access means
   a. Person who developing progamme
   b. Tools for simplifying the programme usage
   c. Relating between tables
   d. Reporting generated by programme.
11. 'Join line' in the context of Access Tables means:
   a. Graphical representation of relationship between tables
   b. Lines bonding the data within table
   c. Line connecting two fields of a table
   d. Line connecting two record of a table.
12. In order to retrieve select data meeting a specified criteria from two different tables of Access database, we may make use of :
   a. Table
   b. Query
   c. Form
   d. Report.

13. To expect a well formatted printable data from Access database, we may use:
    a. Table
    b. Query
    c. Form
    d. Report.

Q2. Answer the following Questions

1. What do you understand by DBMS. Give names of two commarly available DBMS softwares?
2. Differentiate between 'Desktop database' and 'Server database'. List the criteria that may help you in selecting appropriate database?
3. With suitable example illustrate the meaning of 'attributes' as applied to database?
4. Why do we seek to split up in formation into different tables rather than confine it to a single table?
5. What do you understand by terms 'key field' , 'primary key' and 'secondary key' in a database?
6. List the conventions that you will follow, while naming different fields of a table?
7. What are the uses of 'query' object in Access program?
8. What do you understand by 'Form' object in Access and how are they useful?
9. What is the purpose of 'report' object in Access program?
10. What do you understand by database? What are the ways in which data is stored and queried in an Access database?
11. What are the advantages of Access over Excel?
12. Describe in brief the function of 'Table', 'Query', 'Form' and 'Report' object of Access program?

Q3. Skill Review

1. You are starting to plan ahead for you job search. You decide to maintain a database of company information in Access.
2. Search for at least eight companies. You wish to apply for include company name, address, telephone and fax numbers. If possible, also include the name of contact person in their human resorce department.
3. Create job search company safer, accounting database.
4. Open the company information table.
5. Enter at least records for the companies you researched.
6. Adjust column widths, as necessary.
7. Sort the records in ascending order of Name of Company.
8. Preview Table

9. Format all record to a smaller font size.
10. Change the page layout so that entire table fit on the page.
11. Close company information table.
12. Close Job Search Company, acedb database.

ANSWER

Q.1.

1. c. **2.** d. **3.** a. **4.** c **5.** c **6.** c. **7.** a
8. d. **9.** a. **10.** b. **11.** a. **12.** b. **13.** d.